INDEPENDENT VALIDATION OF SENSOR MODELS IN THE COMMUNITY SENSOR MODEL PROGRAM

Henry J. Theiss, Chief Scientist, Photogrammetry
Integrity Applications Incorporated
5180 Parkstone Drive Suite 260
Chantilly, VA 20151
htheiss@integrity-apps.com

Todd E. Johanesen, Chief Scientist
Advanced Capabilities Development Division
National Geospatial-Intelligence Agency
12300 Sunrise Valley Drive MS P-78
Reston, VA 20191
todd.e.johanesen@nga.mil

ABSTRACT

As the military community continues its reliance on imagery from airborne sensors, the need to standardize sensor models for current and future sensors becomes paramount. In support of this requirement, the Community Sensor Model (CSM) program was established within the Department of Defense. Through this program, users of airborne data are provided a common interface to all essential photogrammetric functions while maintaining proprietary aspects of the sensor, through the use of shared libraries. To date, sensor model validation has ensured that all function calls return values for variables in the correct format, as specified by the Interface Control Document (ICD). The authors propose an enhanced method of validation through the use of photogrammetric processes that quantitatively evaluates a CSM's functionality on mission imagery. To support this, the validation team has developed the Generic Software Exploitation Tool (GSET) that stresses the sensor model's essential capabilities: accurate ground-to-image and image-to-ground transformations, sensor model adjustability to account for possible systematic errors, and full rigorous error propagation. One of the most critical GSET functions proposed by the authors is the photogrammetric resection. Once the sensor model is run through this test, the control and checkpoint residuals resulting from the resection will be quantitatively analyzed for modeling quality and the presence of remaining systematic errors. Results are then reported to the CSM developer as feedback and refinement of CSM development until its final delivery.

INTRODUCTION

Background

In the past, sensor model development for software exploitation tools was not standard; developers each had their own proprietary software regarding the structure of functions and associated interfaces. Tool developers often built their own models, so that many versions of the same sensor model existed across the geopositioning community. This led to disparate exploitation processes across the communities, as sensor model developers were not held to any type of standard development. Users could not be assured of consistent photogrammetric results and their exploitation tools received regular updates each time a new sensor was deployed or current sensors were updated, resulting in a time consuming, expensive and ineffective use of resources. Thus, the Community Sensor Model (CSM) program was begun to standardize the method for procurement and distribution of sensor models across the community, while maintaining the proprietary aspects of sensor model developers.

A CSM compliant sensor model consists of a dynamically linked software library that supports photogrammetric operations on images. The software library works through an Application Program Interface (API) to communicate the coordinate transformation process to the Software Exploitation Tool (SET). The software library for CSM is constructed such that each time it is updated, the model can be added or removed from the SET with minimal impact; i.e., the SET does not need to be compiled each time a sensor model is added or removed. The Technical Requirements Document (TRD) mandates that the CSM undergo verification and validation. Accordingly, the TRD outlines two sets of verification and validation. The first tier of verification and validation is with the CSM and occurs as part of the delivery of the finished product. Then the sensor model is available for third party integration within a SET. After it is implemented within a SET, it undergoes verification and validation as a complete system. The final validation of the CSM-enabled SET is typically through an independent party who ensures that any SET, which internally makes any CSM function calls such as `groundToImage` and `computeSensorPartials`, produces output with errors that are consistent with their associated uncertainty estimates.

To date, CSM verification testing software merely ensures that the sensor model meets the requirements set forth by the API. The thrust of this paper is to build a testing tool that verifies that the sensor model adheres to rigorous photogrammetric requirements. In order to focus on this aspect of CSM verification and validation, photogrammetrists – working with in-house software developers – created the Generic SET (GSET). A model that successfully passes the GSET tests exhibits certain characteristics as a minimum: `imageToGround` and `groundToImage` represent accurate transformations relating the image to the 3D world, the model provides for rigorous error propagation, and it provides for refinement of support data using adjustable parameters. Testing against the GSET allows photogrammetrists to identify issues with the sensor model prior to vendors submitting their CSM-compliant sensor model for validation within any other SET. The testing process may include frequent technical exchange meetings with the developer, including the review of design and math documentations and participation in developer verification. This way, sensor model developers have time to make refinements to the sensor model before the final delivery.

CSM Core Photogrammetric Functions

The objective of the tests discussed in this paper is to ensure that the core fundamental functions that are used to perform rigorous photogrammetric processes are working correctly. These four functions include: `groundToImage`, `imageToGround`, `computeGroundPartials`, and `computeSensorPartials`. Note that the first two of these functions will perform error propagation, thus returning an output covariance matrix, if required input covariance matrices are provided.

The `groundToImage` function transforms the X, Y, Z Earth Centered Earth Fixed (ECEF) Cartesian Coordinates of an object point to its associated line and sample image coordinates. Its error form allows for the input of a full 3 by 3 covariance matrix associated with its ECEF Coordinates, and outputs a full 2 by 2 covariance matrix associated with its image coordinates. A full covariance matrix expressing the uncertainty of the sensor parameters is inherently an input, as assigned during the process of sensor model instantiation. Throughout the remainder of this paper, the two forms of this function are abbreviated as g2i and g2iEP, respectively.

The `imageToGround` function computes the X, Y, Z ECEF coordinates of an object point as a function of its line and sample image coordinates and height above the Earth ellipsoid. Its error form allows for the input of a full 2 by 2 covariance matrix associated with the image coordinates and a variance associated with the object height, and outputs a full 3 by 3 covariance matrix associated with the computed object coordinates. As in the g2iEP case, a full sensor parameter covariance matrix is inherently assigned. These two forms of this function are abbreviated as i2g and i2gEP, respectively.

The `computeGroundPartials` function computes the elements of a 2 by 3 matrix of partial derivatives of the line and sample image coordinates with respect to the X, Y, Z ECEF object coordinates. The `computeSensorPartials` function computes the partial derivatives of the line and sample image coordinates with respect to any sensor parameter. Each of these partial derivative functions requires the input of the X, Y, Z ECEF object coordinates. These two functions are abbreviated as CGP and CSP, respectively.

VALIDATION TESTS

The GSET photogrammetric validation tests fall into two basic categories, *consistency testing* and *physical model testing*. The objective of the first set of tests is to verify that all four key photogrammetric functions are internally consistent, while the second set of tests verifies that the sensor model has been implemented rigorously in that it accurately and reliably models the imaging of the real three-dimensional world. Consistency testing requires instantiation of a sensor model, but does not require any real ground data. However, physical model testing requires real three-dimensional ground control data associated with a real image.

Consistency testing includes Verify Closure, Verify Ground Partial Derivatives, Verify Sensor Partial Derivatives, and Verify Error Propagation. Physical model testing includes Quantify Un-modeled Sensor Error and Verify Support Data Quality.

Note that if a sensor model developer provided object oriented software code that performs an i2g function, then any developer could write software to indirectly implement the associated g2i function, which would require iteration since it is a nonlinear solution. Similarly, the associated CSP and CGP functions would be implemented by exercising numerical partial derivatives. In practice, the developer has the choice of whether or not to write the g2i, CSP, and CGP functions such that they internally call the i2g function. While doing so would reduce the amount of analytical work performed by the sensor model developer, it may result in longer computation time for g2i and in loss of precision or instability for the partial derivative computations.

Verify Closure

This is a two part test that verifies consistency between i2g and g2i, and between i2gEP and g2iEP. A closure test is run for every pixel in a given image. At each pixel location (x,y), an associated height above ellipsoid, h, is randomly generated with a uniform distribution such that it falls between the maximum and minimum allowable heights in the ground scene. The i2g function operates on the (x,y,h) triple to yield an (X,Y,Z) ground coordinate in the ECEF Cartesian System. This (X,Y,Z) ground coordinate is then input to the g2i function to compute its (x,y) image coordinates. The input and output (x,y) coordinates are differenced and statistically evaluated. Engineering judgement, which takes into account the physics of the sensor, is applied to determine whether these differences are acceptable. For example, a frame sensor model, which contains a single camera perspective center, would be held to a higher standard than a pushbroom sensor model, which contains time-varying perspective center and pointing directions that may fluctuate due to air turbulence; i.e., in an extreme case the same ground point could project to more than one image location. Figure 1a is a flowchart that illustrates the implementation of the first part of the closure test.

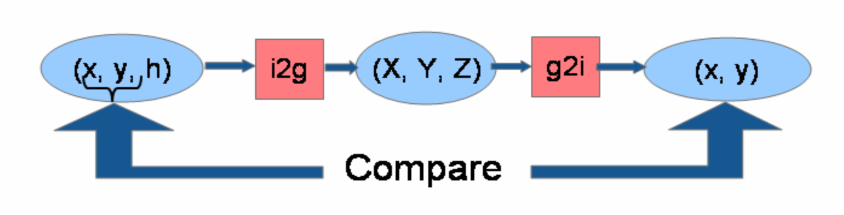


Figure 1a. Closure test for i2g and g2i.

As in the case of the geopositioning closure test, an error propagation closure test is run on all pixels in the image. The objective of this test is to investigate whether the sequential application of i2gEP followed by g2iEP results in a 2 by 2 output covariance matrix of the (x,y) image coordinates with values that are essentially equal to those of its input covariance matrix. As shown in Figure 1b, a 2 by 2 covariance matrix and a variance of the ground height are input to the i2gEP, yielding a full 3 by 3 covariance matrix of the (X,Y,Z) ground coordinates. Then, this full 3 by 3 covariance matrix is input to the g2iEP that returns the elements of a 2 by 2 covariance matrix of the (x,y) image coordinates. Note that for this test, the values of the sensor parameter covariance must be set to essentially zeros in order to eliminate their influence on the results.

If the sensor model corrects for all systematic effects, then the model propagates only the input uncertainties – height variance and image point mensuration covariance – to the ground. However, if there is un-modeled sensor error (see the section entitled “Verify Un-modeled Sensor Error”), then the i2gEP and g2iEP will effect an increase in the values of the covariance values internally. Accommodating un-modeled sensor will cause the values of the covariance output from the closure test to be larger than those of the input covariance, and the magnitude of these differences should be consistent with the values of the 2 by 2 covariance matrix output from the `getUnmodeledError` CSM function.

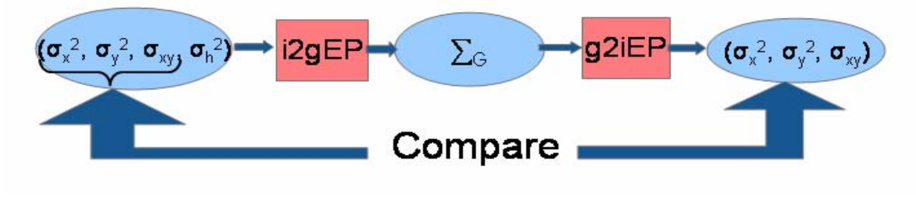

$(\sigma_x^2, \sigma_y^2, \sigma_{xy}, \sigma_h^2)$
i2gEP
Σ_G
g2iEP
$(\sigma_x^2, \sigma_y^2, \sigma_{xy})$
Compare

Figure 1b. Closure test for i2gEP and g2iEP.

Verify Ground Partial Derivatives

Partial derivatives are critical ingredients to allow for solution of the values of unknown variables, and to perform rigorous error propagation. Therefore, to validate that the partial derivatives are working correctly, we generate error-free ground and image coordinate data that correspond perfectly according to the physical sensor model, as implemented by the sensor model developer. Then, we perturb the values of the parameters that correspond to the partial derivatives to be tested, and show that use of these partial derivatives in a least squares adjustment will pull the perturbed parameter values back to their known error-free values. If the adjustment fails to converge or yield the correct results, then applying the definition of the derivative can be used to help diagnose which partial derivative may be in error. An alternate diagnostic technique, to isolate the incorrect partial derivative(s), would be to solve for one variable at a time.

For the case where two overlapping images are available, a two-image ground point intersection is an ideal test to verify that the ground partial derivatives are working correctly. By starting with a ground coordinate expressed by (X,Y,Z), the g2i function is applied to each of the two images to yield (x,y) image coordinates in each image. After the ground coordinates are perturbed, to simulate initial approximations (X',Y',Z') for a least squares adjustment, the two sets of (x,y) image coordinates form four condition equations as a function of three unknown ground coordinates as follows:

$$\begin{aligned}
F_1 &= x_1 - g2i_{x1}(S_1, X, Y, Z) \\
F_2 &= y_1 - g2i_{y1}(S_1, X, Y, Z) \\
F_3 &= x_2 - g2i_{x2}(S_2, X, Y, Z) \\
F_4 &= y_2 - g2i_{y2}(S_2, X, Y, Z)
\end{aligned} \tag{3}$$

where

S_1, S_2 are vectors of known sensor parameters for images 1 and 2, respectively,

X, Y, Z are the coordinates of the unknown ground point (initially X', Y', Z'), and

x_1, y_1, x_2, y_2 are the error-free image coordinates of the point in images 1 and 2, respectively.

In a least squares solution for the coordinates of the ground point, the vector of corrections is computed as follows:

$$\Delta = \left(B^T B\right)^{-1} B^T f \tag{4}$$

where

B is a 4 by 3 matrix of partial derivatives of the four condition equations with respect to the three unknowns, and consists of the six values returned from CGP for each of the images, and

f is the 4 by 1 misclosure vector, and consists of the values of the negative condition equations calculated at the current values of the ground coordinates.

The *X*, *Y*, *Z* ground coordinates are updated by the vector of corrections, and iterations are repeated until convergence, i.e. the corrections become essentially zeroes. The inability for the least squares solution to converge indicates a problem with the CGP function. If the solution converges, then the computed ground coordinates are compared to the perfectly known coordinates to verify that the correct values were computed. See Figure 2a.

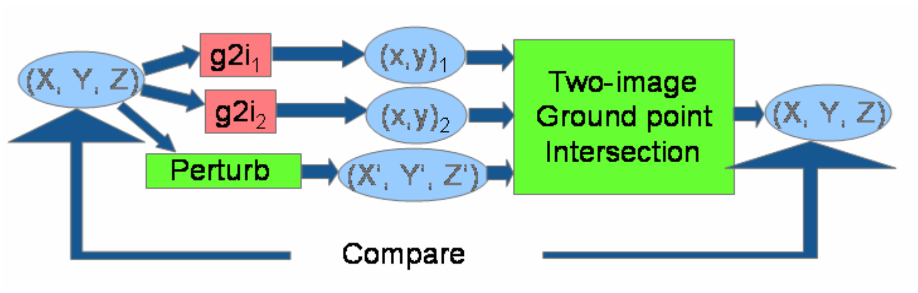


Figure 2a. Verify Ground Partial Derivatives with Two Images.

In many cases overlapping images are difficult, if not impossible, to obtain. Instead of a two-image ground point intersection, a single ray intersection with the inflated ellipsoid can be implemented to verify the values of the ground partial derivatives. The test begins with the latitude, longitude, and height (φ, λ, h) geodetic coordinates of a ground point and converts these to (*X*, *Y*, *Z*) ECEF Geocentric Coordinates. Then, the g2i function is applied to compute the corresponding (x, y) image coordinates. Using perturbed values for the ground coordinates (*X'*, *Y'*, *Z'*) as initial approximations, the (x, y) image coordinates and known height, *h*, are input to a Newton Iteration solution for the unknown ground coordinates. This solution consists of three condition equations and three unknown parameters (*X*, *Y*, *Z*). The first two equations are the first two lines of Equation (3), and the third equation is the equation of an inflated ellipsoid:

$$F_3 = \frac{(X^2 + Y^2)}{(a+h)^2} + \frac{Z^2}{(b+h)^2} - 1 \tag{5}$$

where *a* and *b* are the semi-major and semi-minor axes of the Earth Ellipsoid, respectively. In Newton Iteration the vector of corrections, to be added to the ground coordinates at each iteration, are obtained by:

$$\Delta = B^{-1} f \tag{6}$$

B is a 3 by 3 matrix of partial derivatives of the three condition equations with respect to the three unknowns; i.e., the first two rows consist of the six values returned from CGP and the third row consists of the partial derivatives of Equation (5) with respect to *X*, *Y*, *Z*; and

f is the 3 by 1 misclosure vector, and consists of the values of the negative condition equations evaluated at the current values of the ground coordinates.

As in the two-image ground point intersection, the *X*, *Y*, *Z* ground coordinates are updated by the vector of corrections, and iterations are repeated until convergence. Again, the inability for the Newton Iteration to converge indicates a problem with the CGP function. If the solution converges, then the computed ground coordinates are compared to the perfectly known coordinates to verify that the correct values were computed. See Figure 2b.

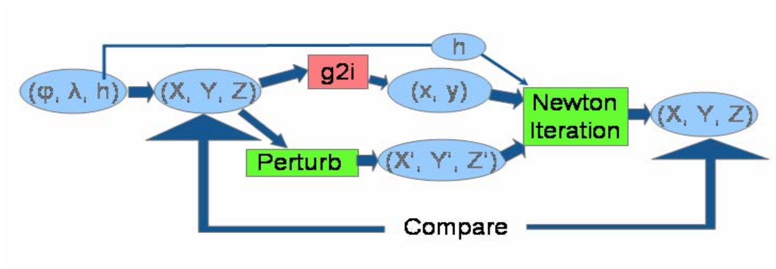


Figure 2b. Verify Ground Partial Derivatives with Single Image.

Verify Sensor Partial Derivatives

As in the verification of ground partial derivatives, the sensor partial derivatives can be evaluated by testing the ability to solve for the correct values of the sensor parameters that have been perturbed from their known values. Solving for the values of some or the entire set of sensor parameters associated with a single image is referred to as photogrammetric resection. We start by first generating a set of error-free data with *n* image points and randomly generated heights, *h*, and their associated ground coordinates (*X*, *Y*, *Z*). Then, we perturb the *u* by 1 vector of sensor parameters, *S*, to give us a vector of initial approximations, *S*'. Using the ground and image coordinates of all points and vector of initial approximations for the values of the sensor parameters, a least squares solution is implemented to solve for the values of the sensor parameters. These values are compared to their original values to verify that the sensor partial derivatives are accurate. The 2*n* condition equations, that are a function of *n* ground points and associated image points used in this least squares solution, are:

$$\begin{aligned} F_1 &= x_1 - g2i_x(S, X_1, Y_1, Z_1) \\ F_2 &= y_1 - g2i_y(S, X_1, Y_1, Z_1) \\ &\vdots \\ F_{2n-1} &= x_n - g2i_x(S, X_n, Y_n, Z_n) \\ F_{2n} &= y_n - g2i_y(S, X_n, Y_n, Z_n) \end{aligned} \tag{7}$$

where

S is the *u* by 1 vector of unknown sensor parameters,
X_i, Y_i, Z_i are the coordinates of the i^{th} known ground point, and
x_i, y_i are the image coordinates of the i^{th} point.

The vector of corrections, to be added to the current values of the sensor parameters at each iteration, is obtained by Equation (4) where:

B is a 2*n* by *u* matrix of partial derivatives of Equations (7) with respect to the unknown sensor parameters; i.e., associated with every point are two rows consisting of the 2*u* values returned from CSP; and
f is the 2*n* by 1 misclosure vector, and consists of the values of the negative condition equations evaluated at the current values of the sensor parameters.

As in the ground point intersection, the *u* elements of the sensor parameter vector, *S*, are updated by the vector of corrections, and iterations are repeated until convergence. The inability for the least squares adjustment to converge would indicate either a problem with the CSP function, or an attempt to solve for too many sensor parameters that may be too highly correlated. It may be necessary to run this test in multiple steps, ensuring that in each step the set of parameters solved for in the adjustment are chosen judiciously. If the solution converges, then the computed sensor parameters are compared to the perfectly known sensor parameters to verify that the correct values were computed. See Figure 3, a flowchart that illustrates how to run this test.

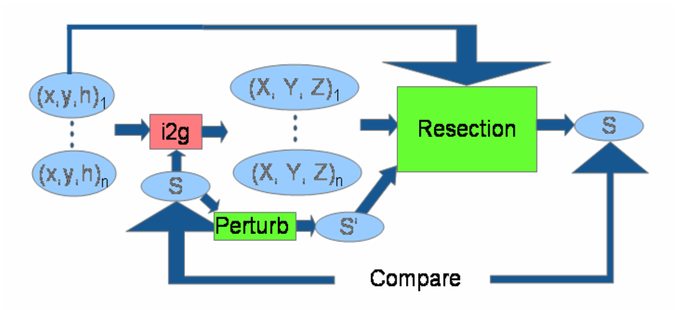


Figure 3. Verify Sensor Partial Derivatives Test.

Verify Error Propagation

The CSM standard requires the sensor model builder to allow for the overload of the g2i and i2g functions such that they perform error propagation; we call these overloaded functions g2iEP and i2gEP, respectively. Since the partial derivative functions, CSP and CGP, are already provided as per the CSM standard, application of error propagation principles (see Mikhail 1976) yields output covariance matrices for the g2i and i2g functions.

The following equation, essentially the first two lines of Equation (3), expresses g2i:

$$\begin{bmatrix} x \\ y \end{bmatrix} = g2i(S, X, Y, Z) \tag{8}$$

Then the 2 by 2 covariance matrix associated with the image coordinates, *x* and *y*, on the left hand side can be computed by applying direct error propagation as a summation as follows:

$$\Sigma_{xy} = J_{IS}\Sigma_S J_{IS}{}^T + J_{IG}\Sigma_G J_{IG}{}^T \tag{9}$$

where

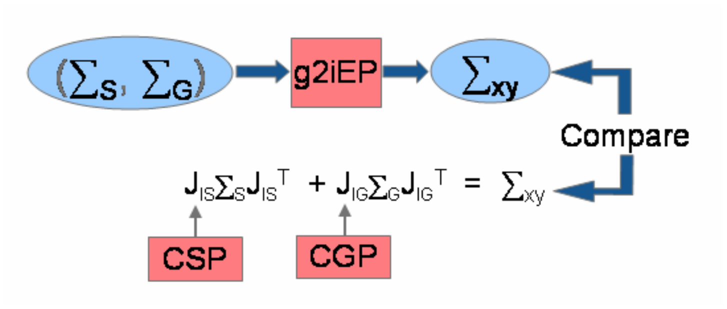
J_{IS} is a 2 by *u* matrix of partial derivatives of Equation (8) with respect to the *u* elements of sensor parameter vector *S*, and consists of the values returned from the CSP function,

J_{IG} is a 2 by 3 matrix of partial derivatives of Equation (8) with respect to the 3 ground coordinates (*X*, *Y*, *Z*), and consists of the values returned from the CGP function,

$\sum_S$ is a *u* by *u* input covariance matrix associated with the vector *S*, and

$\sum_G$ is a 3 by 3 input covariance matrix associated with (*X*, *Y*, *Z*).

Note that Equation (9) is only valid when the sensor parameter vector *S* is uncorrelated with the ground coordinates (*X*, *Y*, *Z*). The values computed by Equation (9) should match closely with those returned from the g2iEP function. However, in the case when the `getUnmodeledError` function returns non-zero values, the 2 by 2 covariance matrix computed by Equation (9) should be increased as such. See Figure 4a, a flowchart illustrating how to verify g2iEP.



(Σ_S, Σ_G)
g2iEP
Σ_{xy}
Compare
$J_{IS}\Sigma_S J_{IS}^T + J_{IG}\Sigma_G J_{IG}^T = \Sigma_{xy}$
CSP
CGP

Figure 4a. Verify g2iEP.

Simple application of error propagation in the form of Equation (9) is not possible to compute the ground coordinate covariance matrix associated with the i2g function, because the ground height, *h*, is an additional input and the equation of the inflated ellipsoid is a necessary third condition equation. So the three condition equations that allow for the iterative calculation of i2g include the two components of Equation (8) and Equation (5).

Then the 3 by 3 covariance matrix associated with the ground coordinates (*X*, *Y*, *Z*) can be computed by error propagation (see Mikhail 1976) as follows:

$$\Sigma_G = \left(B^T \left(A\Sigma A^T\right)^{-1} B\right)^{-1} \tag{10}$$

where

$$\Sigma = \begin{bmatrix} \Sigma_{xy} & & \\ & \sigma_h^{\,2} & \\ & & \Sigma_S \end{bmatrix}, \text{ a block diagonal matrix,}$$

A is a 3 by (3+*u*) matrix of partial derivatives of the 3 condition equations with respect to the scalars *x*, *y*, *h* and elements of vector *S*,

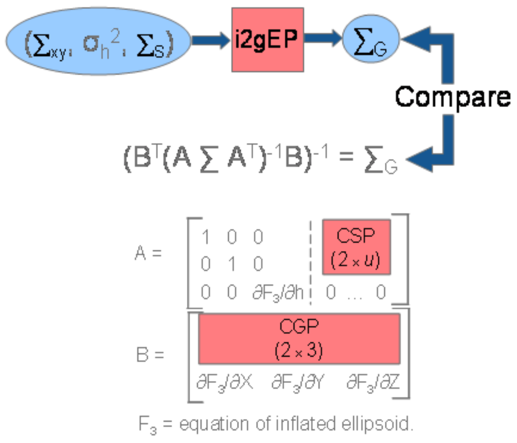
B is a 3 by 3 matrix of partial derivatives of the 3 condition equations with respect to the ground coordinates *X*, *Y*, *Z*,

$\sum_{xy}$ is a 2 by 2 input covariance matrix associated with the (*x*, *y*) image coordinates, influenced by the values returned by the `getUnmodeledError` function, if applicable,

$\sigma_h^{\,2}$ is the input variance of the height of the ground point above ellipsoid, and

$\sum_S$ is a *u* by *u* input covariance matrix associated with the sensor parameters.

The values of the covariance matrix computed by Equation (10) should match closely with those returned from the i2gEP function. See Figure 4b for a flowchart illustrating the i2gEP verification procedure; note specifically how the *A* and *B* matrices consist of values returned from the CSP and CGP functions, respectively.



$(\Sigma_{xy}, \sigma_h^2, \Sigma_S)$
i2gEP
Σ_G
Compare
$(B^T(A \Sigma A^T)^{-1}B)^{-1} = \Sigma_G$
A =
CSP (2 × u)
$\partial F_3/\partial h$
B =
CGP (2 × 3)
$\partial F_3/\partial X$ $\partial F_3/\partial Y$ $\partial F_3/\partial Z$
F_3 = equation of inflated ellipsoid.

Figure 4b. Verify i2gEP.

Quantify Un-modeled Sensor Error

The objective of quantifying un-modeled sensor error is to obtain a measure of the internal geometric fidelity of the sensor model. One way to assess internal geometry is to compute the root mean square (RMS) error of check point residuals after control points have been used to perform a resection of the image, where the optimization function to be minimized is the weighted sum of the squared image coordinate residuals. Evaluating the relative accuracy between pairs of points, after performing the resection, is another way to assess the metric quality of a sensor model.

The resection used in this test requires the manual measurement of image coordinates associated with a large number of ground points with accurately known *X*, *Y*, *Z* coordinates. These points would ideally span the entire image in a relatively dense pattern. Typically around half of the points are used as control in the resection, and the check points withheld from the solution are used to assess the remaining sensor model error. See Figure 5, a flowchart illustrating the resection based procedure involving analysis of residuals.

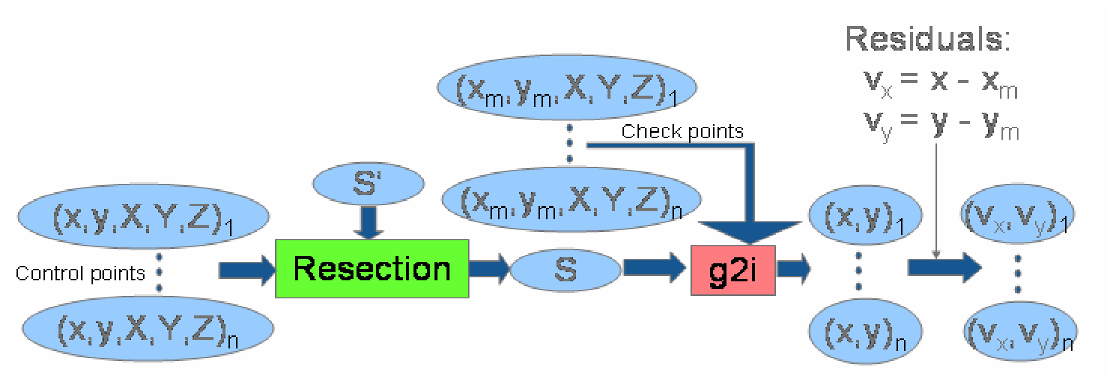


Figure 5. Test to Quantify Un-modeled Sensor Error.

Analysis proceeds with computation of the 2 by 2 image space covariance matrix associated with the check point residuals. Plotting these residuals at their measured locations in image space aids the visual analysis and ability to diagnose un-modeled systematic errors. At a higher level, the computed covariance of the check point residuals should be consistent with the sum of the average image mensuration covariance matrix and the average covariance matrix returned from the `getUnmodeledError` function. Or, on a point by point basis, each check point can be run through the g2iEP to obtain uncertainties, expressed by a 2 by 2 covariance matrix, of the image point in the image plane; and the actual residuals can be compared to these propagated uncertainties. Taking the analysis to another level, error propagation can be applied to compute a 2 by 2 covariance matrix expressing the relative uncertainty between any pair of points in the scene, and a comparison can be made between the actual measured image space discrepancies between a pair of points and their uncertainties predicted by error propagation. The error propagation associated with relative error prediction needs to take into account the values returned from the CSM `getUnmodeledCrossCovariance` function, since unmodeled errors may impact a pair of points in a similar way. For example, analysis can assess whether the relative error between computed coordinates of a pair of points is correlated to the distance between them, and whether the error propagation models this relationship.

Many of the images of interest to the CSM community are collected without the intent of exploiting stereo geometry. Hence, the focus of these tests has been resections instead of block triangulation. So, in cases where overlapping images with significantly large convergence angles are available, we recommend extending the Quantify Un-modeled Sensor Error test from a resection to a multiple image block triangulation. The check point analysis would then assess the quality of three-dimensional object reconstruction via 3D RMS, e.g. discrepancies in the East, North, and Up directions.

Verify Support Data Quality

The developer of a sensor model is responsible for working with the sensor builder to ensure that the support data that accompanies an image is consistent and reliable. In this paper the term *support data quality* does not necessarily mean that the values expressing the sensor position and pointing direction, for example, are highly accurate. Some imaging systems will be limited in their geopositioning accuracy due to relatively inferior quality GPS and inertial measuring unit (IMU), or time synchronization issues, for example. However, the support data quality may still be considered acceptable if the associated covariance information is consistent with the geopositioning accuracy capabilities of the system.

The purpose of the Verify Support Data Quality test is to measure the geopositioning errors on the ground that are obtained from i2gEP when the original support data is used; i.e., no resection or triangulation has been performed to improve the accuracy of the values of the sensor parameters. Since i2gEP will provide a ground coordinate covariance matrix, the geopositioning errors can be compared to the uncertainty estimates to check for reliability. It can then be determined whether there is a direct relationship between increasing errors and increasing uncertainty estimates, and whether the uncertainty estimates as a whole are either too conservative or overly optimistic. Figure 6a shows how the horizontal geopositioning errors can be plotted versus their associated CE90 (circular error, in the horizontal plane, of 90% probability) values, which can be extracted from the 3 by 3 ground covariance matrix output from i2gEP. A similar plot can be made of vertical errors versus LE90 (linear error, in the vertical direction, of 90% probability). Note that to be meaningful, this test requires a large number of data points. Since multiple points appearing on the same image are often highly correlated, it is more beneficial in this test to acquire a few points on each of a very large number of images taken from varying geometries throughout the world. It is not useful to have very few images with a large number of points, as was acceptable for the Quantify Un-modeled Sensor Error test.

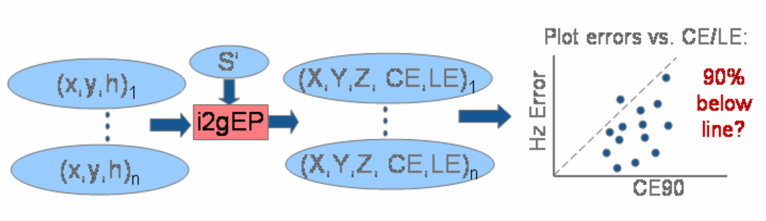


Figure 6a. Verify Support Data Quality.

Performing the analysis of support data quality can provide insight into un-modeled systematic errors or biases caused by a deficiency in the sensor model. For example, a bore-sight misalignment may go unnoticed when inspecting residuals for a single image, due to dominating effects such as pointing error. However, when a large sample of imagery is studied, the bias may become more apparent, especially when the residuals are plotted in image space against axes that correspond to directions impacted by a bore-sight misalignment. Viewing residuals in this manner may also assist in diagnosis of other sensor modeling issues such as neglecting atmospheric refraction. Figure 6b illustrates the strategy for inspecting residuals to potentially identify a bias that may imply a sensor modeling deficiency.

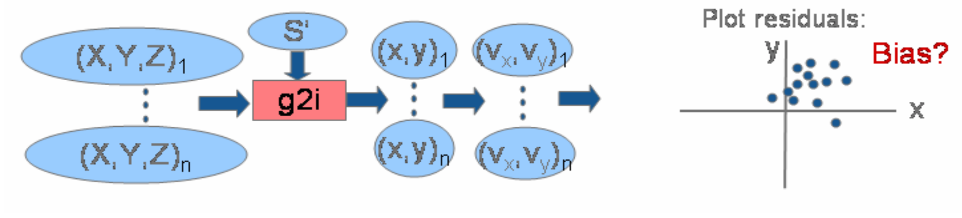


Figure 6b. Investigate Systemmatic Errors.

RESULTS OF A REAL SENSOR MODEL EVALUATION

Over the past two years, the authors have participated in CSM technical exchange meetings, and have recently begun a collaborative effort with a sensor model developer to build the GSET in tandem with evaluating their sensor model. Testing has begun on an initial delivery of the sensor model that is known by the developer to be incomplete at this point. The authors and developer agreed that this is a work-in-progress that can benefit from collaboration in order to reach its highest quality. Further, we agreed that delivering an intermediate sensor model at an earlier stage of the process would benefit the overall product by detecting issues at the earliest time possible. Also, it has allowed the authors to do a better job at developing the GSET.

The testing falls into two main categories as defined earlier in this paper: consistency testing and physical model testing. The real sensor model decisively passed the first three of four consistency tests: Verify Closure, Verify Ground Partial Derivatives, and Verify Sensor Partial Derivatives. The Verify Error Propagation test could not be run since the i2gEP and g2iEP functions were still under development, and for the same reason the second part of the Closure test could not be run. We did, however, identify an issue with the g2i function returning unreasonable default values for the *x* and *y* image coordinates when their values fell outside of the bounds of the image. In order to run the other GSET tests successfully, the authors needed to use weighted least squares to temporarily de-weight the points that were being assigned the unreasonable coordinate values until they fell back inside of the image bounds.

While consistency testing indicated strong performance of internal consistency for the sensor model, physical model testing indicated some potentially serious issues with the model. The sensor model appears to have an inconsistency due to either internal geometry misrepresentation or poor quality support data. Significantly large control point residuals followed a pattern that appeared to be related to the support data variations; however, geometric distortions did not appear to be present in the scene. Further investigation between the authors and developer indicated that some critical support data components are not available. The authors and developer have begun collaboration to attempt to correct for the inconsistencies using additional adjustable parameters. Any remaining sensor modeling error needs to be reflected in the values returned from the `getUnmodeledError` function.

CONCLUSIONS

The CSM standard defines an interface to which sensor model developers can write their software. Photogrammetric exploitation tools benefit from object-oriented code that allows for plug-and-play insertion of sensor model software libraries developed by any third party. The authors designed a GSET to perform independent validation tests to ensure that the sensor model built to adhere to interface standards also adheres to the most stringent photogrammetric standards. A real sensor was used as an example of how sensor builders, who have the most knowledge about the physics of their sensor, can collaborate with photogrammetrists to build the capabilities for image-to-ground transformations, rigorous error propagation, and sensor parameter adjustability that are the key capabilities required to perform rigorous photogrammetric processes.

ACKNOWLEDGMENTS

The authors acknowledge with sincere thanks the technical and software support provided by colleagues Mr. Aaron Braun and Mr. Adam Gurson, respectively. The work performed by the authors was funded by the US Government. The authors also truly appreciate the consistent cooperation of the developer who built the sensor model, and most importantly their firm commitment to providing a reliable sensor model to the customer.

REFERENCES

Mikhail, E., J. Bethel, J. McGlone (2001). Introduction to Modern Photogrammetry. John Wiley and Sons, Inc.
Mikhail, E. (1976). Observations and Least Squares, University Press of America, New York, NY.